Journal of Machine Learning Research 13 (2012) 1-30 Submitted 2/11; Revised 1/12; Published XX/12

Positive Semidefinite Metric Learning Using Boosting-like Algorithms

Chunhua Shen CHUNHUA.SHEN@ADELAIDE.EDU.AU
The University of Adelaide, Adelaide, SA 5005, Australia

Junae Kim JUNAE.KIM@NICTA.COM.AU
NICTA, Canberra Research Laboratory, Locked Bag 8001, Canberra, ACT 2601, Australia

Lei Wang LEIW@UOW.EDU.AU
University of Wollongong, Wollongong, NSW 2522, Australia

Anton van den Hengel ANTON.VANDENHENGEL@ADELAIDE.EDU.AU
The University of Adelaide, Adelaide, SA 5005, Australia

Editor: Sören Sonnenburg, Francis Bach, Cheng Soon Ong

Abstract

The success of many machine learning and pattern recognition methods relies heavily upon the identification of an appropriate distance metric on the input data. It is often beneficial to learn such a metric from the input training data, instead of using a default one such as the Euclidean distance. In this work, we propose a boosting-based technique, termed BOOSTMETRIC, for learning a quadratic Mahalanobis distance metric. Learning a valid Mahalanobis distance metric requires enforcing the constraint that the matrix parameter to the metric remains positive semidefinite. Semidefinite programming is often used to enforce this constraint, but does not scale well and is not easy to implement. BOOSTMETRIC is instead based on the observation that any positive semidefinite matrix can be decomposed into a linear combination of trace-one rank-one matrices. BOOSTMETRIC thus uses rank-one positive semidefinite matrices as weak learners within an efficient and scalable boosting-based learning process. The resulting methods are easy to implement, efficient, and can accommodate various types of constraints. We extend traditional boosting algorithms in that its weak learner is a positive semidefinite matrix with trace and rank being one rather than a classifier or regressor. Experiments on various datasets demonstrate that the proposed algorithms compare favorably to those state-of-the-art methods in terms of classification accuracy and running time.

Keywords: Mahalanobis distance, semidefinite programming, column generation, boosting, Lagrange duality, large margin nearest neighbor.

1. Introduction

The identification of an effective metric by which to measure distances between data points is an essential component of many machine learning algorithms including k-nearest neighbor (kNN), k-means clustering, and kernel regression. These methods have been applied to a range of problems, including image classification and retrieval (Hastie and Tibshirani, 1996; Yu et al., 2008; Jian and Vemuri, 2007; Xing et al., 2002; Bar-Hillel et al., 2005; Boiman et al., 2008; Frome et al., 2007) amongst a host of others.

The Euclidean distance has been shown to be effective in a wide variety of circumstances. Boiman et al. (2008), for instance, showed that in generic object recognition with local features,

*k*NN with a Euclidean metric can achieve comparable or better accuracy than more sophisticated classifiers such as support vector machines (SVMs). The Mahalanobis distance represents a generalization of the Euclidean distance, and offers the opportunity to learn a distance metric directly from the data. This learned Mahalanobis distance approach has been shown to offer improved performance over Euclidean distance-based approaches, and was particularly shown by Wang et al. (2010b) to represent an improvement upon the method of Boiman et al. (2008). It is the prospect of a significant performance improvement from fundamental machine learning algorithms which inspires the approach presented here.

If we let $\mathbf{a}_i, i = 1, 2 \cdots$, represent a set of points in $\mathbb{R}^D$, then the Mahalanobis distance, or Gaussian quadratic distance, between two points is

$$\|\mathbf{a}_i - \mathbf{a}_j\|_{\mathbf{X}} = \sqrt{(\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{X} (\mathbf{a}_i - \mathbf{a}_j)}, \tag{1}$$

where $\mathbf{X} \succcurlyeq 0$ is a positive semidefinite (p.s.d.) matrix. The Mahalanobis distance is thus parameterized by a p.s.d. matrix, and methods for learning Mahalanobis distances are therefore often framed as constrained semidefinite programs. The approach we propose here, however, is based on boosting, which is more typically used for learning classifiers. The primary motivation for the boosting-based approach is that it scales well, but its efficiency in dealing with large data sets is also advantageous. The learning of Mahalanobis distance metrics represents a specific application of a more general method for matrix learning which we present below.

We are interested here in the case where the training data consist of a set of constraints upon the relative distances between data points,

$$\mathcal{I} = \{(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \,|\, \mathbf{dist}_{ij} < \mathbf{dist}_{ik}\}, \tag{2}$$

where $\mathbf{dist}_{ij}$ measures the distance between $\mathbf{a}_i$ and $\mathbf{a}_j$. Each such constraint implies that "$\mathbf{a}_i$ is closer to $\mathbf{a}_j$ than $\mathbf{a}_i$ is to $\mathbf{a}_k$". Constraints such as these often arise when it is known that $\mathbf{a}_i$ and $\mathbf{a}_j$ belong to the same class of data points while $\mathbf{a}_i, \mathbf{a}_k$ belong to different classes. These comparison constraints are thus often much easier to obtain than either the class labels or distances between data elements (Schultz and Joachims, 2003). For example, in video content retrieval, faces extracted from successive frames at close locations can be safely assumed to belong to the same person, without requiring the individual to be identified. In web search, the results returned by a search engine are ranked according to the relevance, an ordering which allows a natural conversion into a set of constraints.

The problem of learning a p.s.d. matrix such as $\mathbf{X}$ can be formulated in terms of estimating a projection matrix $\mathbf{L}$ where $\mathbf{X} = \mathbf{L}\mathbf{L}^\top$. This approach has the advantage that the p.s.d. constraint is enforced through the parameterization, but the disadvantage is that the relationship between the distance measure and the parameter matrix is less direct. In practice this approach has lead to local, rather than globally optimal solutions, however (see (Goldberger et al., 2004) for example).

Methods such as (Xing et al., 2002; Weinberger et al., 2005; Weinberger and Saul, 2006; Globerson and Roweis, 2005) which seek $\mathbf{X}$ directly are able to guarantee global optimality, but at the cost of a heavy computational burden and poor scalability as it is not trivial to preserve the semidefiniteness of $\mathbf{X}$ during the course of learning. Standard approaches such as interior-point (IP) Newton methods need to calculate the Hessian. This typically requires $O(D^4)$ storage and has worst-case computational complexity of approximately $O(D^{6.5})$ where D is the size of the p.s.d. matrix. This is prohibitive for many real-world problems. An alternating projected (sub-)gradient approach is

adopted in (Weinberger et al., 2005; Xing et al., 2002; Globerson and Roweis, 2005). The disadvantages of this algorithm, however, are: 1) it is not easy to implement; 2) many parameters are involved; 3) usually it converges slowly.

We propose here a method for learning a p.s.d. matrix labeled BOOSTMETRIC. The method is based on the observation that any positive semidefinite matrix can be decomposed into a linear positive combination of trace-one rank-one matrices. The weak learner in BOOSTMETRIC is thus a trace-one rank-one p.s.d. matrix. The proposed BOOSTMETRIC algorithm has the following desirable properties:

1. BOOSTMETRIC is efficient and scalable. Unlike most existing methods, no semidefinite programming is required. At each iteration, only the largest eigenvalue and its corresponding eigenvector are needed.

2. BOOSTMETRIC can accommodate various types of constraints. We demonstrate the use of the method to learn a Mahalanobis distance on the basis of a set of proximity comparison constraints.

3. Like AdaBoost, BOOSTMETRIC does not have any parameter to tune. The user only needs to know when to stop. Also like AdaBoost it is easy to implement. No sophisticated optimization techniques are involved. The efficacy and efficiency of the proposed BOOSTMETRIC is demonstrated on various datasets.

4. We also propose a totally-corrective version of BOOSTMETRIC. As in TotalBoost (Warmuth et al., 2006) the weights of all the selected weak learners (rank-one matrices) are updated at each iteration.

   Both the stage-wise BOOSTMETRIC and totally-corrective BOOSTMETRIC methods are very easy to implement.

The primary contributions of this work are therefore as follows: 1) We extend traditional boosting algorithms such that each weak learner is a matrix with the trace and rank of one—which must be positive semidefinite—rather than a classifier or regressor; 2) The proposed algorithm can be used to solve many semidefinite optimization problems in machine learning and computer vision. We demonstrate the scalability and effectiveness of our algorithms on metric learning. Part of this work appeared in Shen et al. (2008, 2009). More theoretical analysis and experiments are included in this version. Next, we review some relevant work before we present our algorithms.

1.1 Related Work

Distance metric learning is closely related to subspace methods. Principal component analysis (PCA) and linear discriminant analysis (LDA) are two classical dimensionality reduction techniques. PCA finds the subspace that captures the maximum variance within the input data while LDA tries to identify the projection which maximizes the between-class distance and minimizes the within-class variance. Locality preserving projection (LPP) finds a linear projection that preserves the neighborhood structure of the data set (He et al., 2005). Essentially, LPP linearly approximates the eigenfunctions of the Laplace Beltrami operator on the underlying manifold. The connection between LPP and LDA is also revealed in (He et al., 2005). Wang et al. (2010a) extended LPP to supervised multi-label classification. Relevant component analysis (RCA) (Bar-Hillel et al., 2005)

learns a metric from *equivalence* constraints. RCA can be viewed as extending LDA by incorporating must-link constraints and cannot-link constraints into the learning procedure. Each of these methods may be seen as devising a linear projection from the input space to a lower-dimensional output space. If this projection is characterized by the matrix $\mathbf{L}$, then note that these methods may be related to the problem of interest here by observing $\mathbf{X} = \mathbf{L}\mathbf{L}^\top$. This typically implies that $\mathbf{X}$ is rank-deficient.

Recently, there has been significant research interest in supervised distance metric learning using side information that is typically presented in a set of pairwise constraints. Most of these methods, although appearing in different formats, share a similar essential idea: to learn an optimal distance metric by keeping training examples in equivalence constraints close, and at the same time, examples in in-equivalence constraints well separated. Previous work of (Xing et al., 2002; Weinberger et al., 2005; Jian and Vemuri, 2007; Goldberger et al., 2004; Bar-Hillel et al., 2005; Schultz and Joachims, 2003) fall into this category. The requirement that $\mathbf{X}$ must be p.s.d. has led to the development of a number of methods for learning a Mahalanobis distance which rely upon constrained semidefinite programing. This approach has a number of limitations, however, which we now discuss with reference to the problem of learning a p.s.d. matrix from a set of constraints upon pairwise-distance comparisons. Relevant work on this topic includes (Bar-Hillel et al., 2005; Xing et al., 2002; Jian and Vemuri, 2007; Goldberger et al., 2004; Weinberger et al., 2005; Globerson and Roweis, 2005) amongst others.

Xing et al. (2002) first proposed the idea of learning a Mahalanobis metric for clustering using convex optimization. The inputs are two sets: a similarity set and a dis-similarity set. The algorithm maximizes the distance between points in the dis-similarity set under the constraint that the distance between points in the similarity set is upper-bounded. Neighborhood component analysis (NCA) (Goldberger et al., 2004) and large margin nearest neighbor (LMNN) (Weinberger et al., 2005) learn a metric by maintaining consistency in data's neighborhood and keep a large margin at the boundaries of different classes. It has been shown in (Weinberger and Saul, 2009; Weinberger et al., 2005) that LMNN delivers the state-of-the-art performance among most distance metric learning algorithms. Information theoretic metric learning (ITML) learns a suitable metric based on information theoretics (Davis et al., 2007). To partially alleviate the heavy computation of standard IP Newton methods, Bregman's cyclic projection is used in Davis et al. (2007). This idea is extended in Wang and Jin (2009), which has a closed-form solution and is computationally efficient.

There have been a number of approaches developed which aim to improve the scalability of the process of learning a metric parameterized by a p.s.d. metric $\mathbf{X}$. For example, Rosales and Fung (2006) approximate the p.s.d. cone using a set of linear constraints based on the diagonal dominance theorem. The approximation is not accurate, however, in the sense that it imposes too strong a condition on the learned matrix—one may not want to learn a diagonally dominant matrix. Alternative optimization is used in (Xing et al., 2002; Weinberger et al., 2005) to solve the semidefinite problem iteratively. At each iteration, a full eigen-decomposition is applied to project the solution back onto the p.s.d. cone. BOOSTMETRIC is conceptually very different to this approach, and additionally only requires the calculation of the first eigenvector. Tsuda et al. (2005) proposed to use matrix logarithms and exponentials to preserve positive definiteness. For the application of semidefinite kernel learning, they designed a matrix exponentiated gradient method to optimize von Neumann divergence based objective functions. At each iteration of matrix exponentiated gradient, a full eigen-decomposition is needed. In contrast, we only need to find the leading eigenvector.

The approach proposed here is directly inspired by the LMNN proposed in (Weinberger and Saul, 2009; Weinberger et al., 2005). Instead of using the hinge loss, however, we use the exponential loss and logistic loss functions in order to derive an AdaBoost-like (or LogitBoost-like) optimization procedure. In theory, any differentiable convex loss function can be applied here. Hence, despite similar purposes, our algorithm differs essentially in the optimization. While the formulation of LMNN looks more similar to SVMs, our algorithm, termed BOOSTMETRIC, largely draws upon AdaBoost (Schapire, 1999).

Column generation was first proposed by Dantzig and Wolfe (1960) for solving a particular form of structured linear program with an extremely large number of variables. The general idea of column generation is that, instead of solving the original large-scale problem (master problem), one works on a restricted master problem with a reasonably small subset of the variables at each step. The dual of the restricted master problem is solved by the simplex method, and the optimal dual solution is used to find the new column to be included into the restricted master problem. LPBoost (Demiriz et al., 2002) is a direct application of column generation in boosting. Significantly, LPBoost showed that in an LP framework, unknown weak hypotheses can be learned from the dual although the space of all weak hypotheses is infinitely large. Shen and Li (2010) applied column generation to boosting with general loss functions. It is these results that underpin BOOSTMETRIC.

The remaining content is organized as follows. In Section 2 we present some preliminary mathematics. In Section 3, we show the main results. Experimental results are provided in Section 4.

2. Preliminaries

We introduce some fundamental concepts that are necessary for setting up our problem. First, the notation used in this paper is as follows.

2.1 Notation

Throughout this paper, a matrix is denoted by a bold upper-case letter ($\mathbf{X}$); a column vector is denoted by a bold lower-case letter ($\boldsymbol{x}$). The ith row of $\mathbf{X}$ is denoted by $\mathbf{X}_{i:}$ and the ith column $\mathbf{X}_{:i}$. $\mathbf{1}$ and $\mathbf{0}$ are column vectors of 1's and 0's, respectively. Their size should be clear from the context. We denote the space of $D \times D$ symmetric matrices by $\mathbb{S}^D$, and positive semidefinite matrices by $\mathbb{S}^D_+$. $\mathbf{Tr}(\cdot)$ is the trace of a symmetric matrix and $\langle \mathbf{X}, \mathbf{Z} \rangle = \mathbf{Tr}(\mathbf{X}\mathbf{Z}^\top) = \sum_{ij} \mathbf{X}_{ij}\mathbf{Z}_{ij}$ calculates the inner product of two matrices. An element-wise inequality between two vectors like $\boldsymbol{u} \leq \boldsymbol{v}$ means $u_i \leq v_i$ for all i. We use $\mathbf{X} \succcurlyeq 0$ to indicate that matrix $\mathbf{X}$ is positive semidefinite. For a matrix $\mathbf{X} \in \mathbb{S}^D$, the following statements are equivalent: 1) $\mathbf{X} \succcurlyeq 0$ ($\mathbf{X} \in \mathbb{S}^D_+$); 2) All eigenvalues of $\mathbf{X}$ are nonnegative ($\lambda_i(\mathbf{X}) \geq 0$, $i = 1, \cdots, D$); and 3) $\forall \boldsymbol{u} \in \mathbb{R}^D$, $\boldsymbol{u}^\top \mathbf{X} \boldsymbol{u} \geq 0$.

2.2 A Theorem on Trace-one Semidefinite Matrices

Before we present our main results, we introduce an important theorem that serves the theoretical basis of BOOSTMETRIC.

Definition 2.1. For any positive integer m, given a set of points $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_m\}$ in a real vector or matrix space Sp, the *convex hull* of Sp spanned by m elements in Sp is defined as:

$$\mathbf{Conv}_m(\mathrm{Sp}) = \left\{ \sum_{i=1}^m w_i \boldsymbol{x}_i \,\middle|\, w_i \geq 0, \sum_{i=1}^m w_i = 1, \boldsymbol{x}_i \in \mathrm{Sp} \right\}.$$

Define the linear convex span of Sp as:[1]

$$\mathbf{Conv}(\mathrm{Sp}) = \bigcup_m \mathbf{Conv}_m(\mathrm{Sp}) = \left\{ \sum_{i=1}^m w_i \boldsymbol{x}_i \,\middle|\, w_i \geq 0, \sum_{i=1}^m w_i = 1, \boldsymbol{x}_i \in \mathrm{Sp}, m \in \mathbb{Z}_+ \right\}.$$

Here $\mathbb{Z}_+$ denotes the set of all positive integers.

Definition 2.2. Let us define Γ_1 to be the space of all positive semidefinite matrices $\mathbf{X} \in \mathbb{S}_+^D$ with trace equaling one:

$$\Gamma_1 = \{\mathbf{X} \,|\, \mathbf{X} \succcurlyeq 0, \mathbf{Tr}(\mathbf{X}) = 1\};$$

and Ψ_1 to be the space of all positive semidefinite matrices with both trace and rank equaling one:

$$\Psi_1 = \{\mathbf{Z} \,|\, \mathbf{Z} \succcurlyeq 0, \mathbf{Tr}(\mathbf{Z}) = 1, \mathbf{Rank}(\mathbf{Z}) = 1\}.$$

We also define Γ_2 as the convex hull of Ψ_1, *i.e.*,

$$\Gamma_2 = \mathbf{Conv}(\Psi_1).$$

Lemma 2.1. *Let Ψ_2 be a convex polytope defined as $\Psi_2 = \{\boldsymbol{\lambda} \in \mathbb{R}^D | \lambda_k \geq 0, \forall k = 0, \cdots, D, \sum_{k=1}^D \lambda_k = 1\}$, then the points with only one element equaling one and all the others being zeros are the extreme points (vertexes) of Ψ_2. All the other points can not be extreme points.*

Proof. Without loss of generality, let us consider such a point $\boldsymbol{\lambda}' = \{1, 0, \cdots, 0\}$. If $\boldsymbol{\lambda}'$ is not an extreme point of Ψ_2, then it must be possible to express it as a convex combination of a set of *other* points in Ψ_2: $\boldsymbol{\lambda}' = \sum_{i=1}^m w_i \boldsymbol{\lambda}^i$, $w_i > 0$, $\sum_{i=1}^m w_i = 1$ and $\boldsymbol{\lambda}^i \neq \boldsymbol{\lambda}'$. Then we have equations: $\sum_{i=1}^m w_i \lambda_k^i = 0$, $\forall k = 2, \cdots, D$. It follows that $\lambda_k^i = 0$, $\forall i$ and $k = 2, \cdots, D$. That means, $\lambda_1^i = 1$ $\forall i$. This is inconsistent with $\boldsymbol{\lambda}^i \neq \boldsymbol{\lambda}'$. Therefore such a convex combination does not exist and $\boldsymbol{\lambda}'$ must be an extreme point. It is trivial to see that any $\boldsymbol{\lambda}$ that has more than one active element is an convex combination of the above-defined extreme points. So they can not be extreme points. ■

Theorem 2.1. *Γ_1 equals to Γ_2;* i.e., *Γ_1 is also the convex hull of Ψ_1. In other words, all $\mathbf{Z} \in \Psi_1$, form the set of extreme points of Γ_1.*

Proof. It is easy to check that any convex combination $\sum_i w_i \mathbf{Z}_i$, such that $\mathbf{Z}_i \in \Psi_1$, resides in Γ_1, with the following two facts: 1) a convex combination of p.s.d. matrices is still a p.s.d. matrix; 2) $\mathbf{Tr}\left(\sum_i w_i \mathbf{Z}_i\right) = \sum_i w_i \mathbf{Tr}(\mathbf{Z}_i) = 1$.

By denoting $\lambda_1 \geq \cdots \geq \lambda_D \geq 0$ the eigenvalues of a $\mathbf{Z} \in \Gamma_1$, we know that $\lambda_1 \leq 1$ because $\sum_{i=1}^D \lambda_i = \mathbf{Tr}(\mathbf{Z}) = 1$. Therefore, all eigenvalues of $\mathbf{Z}$ must satisfy: $\lambda_i \in [0, 1]$, $\forall i = 1, \cdots, D$ and $\sum_i^D \lambda_i = 1$. By looking at the eigenvalues of $\mathbf{Z}$ and using Lemma 2.1, it is immediate to see that a matrix $\mathbf{Z}$ such that $\mathbf{Z} \succcurlyeq 0$, $\mathbf{Tr}(\mathbf{Z}) = 1$ and $\mathbf{Rank}(\mathbf{Z}) > 1$ can not be an extreme point of Γ_1. The only candidates for extreme points are those rank-one matrices ($\lambda_1 = 1$ and $\lambda_{2,\cdots,D} = 0$). Moreover, it is not possible that some rank-one matrices are extreme points and others are not because the other two constraints $\mathbf{Z} \succcurlyeq 0$ and $\mathbf{Tr}(\mathbf{Z}) = 1$ do not distinguish between different rank-one matrices.

Hence, all $\mathbf{Z} \in \Psi_1$ form the set of extreme points of Γ_1. Furthermore, Γ_1 is a convex and compact set, which must have extreme points. The Krein-Milman Theorem (Krein and Milman, 1940) tells us that a convex and compact set is equal to the convex hull of its extreme points. ■

1. With slight abuse of notation, we also use the symbol $\mathbf{Conv}(\cdot)$ to denote convex span. In general it is not a convex hull.

This theorem is a special case of the results from (Overton and Womersley, 1992) in the context of eigenvalue optimization. A different proof for the above theorem's general version can also be found in (Fillmore and Williams, 1971).

In the context of semidefinite optimization, what is of interest about Theorem 2.1 is as follows: it tells us that a bounded p.s.d. matrix constraint $\mathbf{X} \in \Gamma_1$ can be equivalently replaced with a set of constrains which belong to Γ_2. At the first glance, this is a highly counterintuitive proposition because Γ_2 involves many more complicated constraints. Both w_i and $\mathbf{Z}_i$ ($\forall i = 1, \cdots, m$) are unknown variables. Even worse, m could be extremely (or even infinitely) large. Nevertheless, this is the type of problems that *boosting* algorithms are designed to solve. Let us give a brief overview of boosting algorithms.

2.3 Boosting

Boosting is an example of ensemble learning, where multiple learners are trained to solve the same problem. Typically a boosting algorithm (Schapire, 1999) creates a single strong learner by incrementally adding base (weak) learners to the final strong learner. The base learner has an important impact on the strong learner. In general, a boosting algorithm builds on a user-specified base learning procedure and runs it repeatedly on modified data that are outputs from the previous iterations.

The general form of the boosting algorithm is sketched in Algorithm 1. The inputs to a boosting algorithm are a set of training example $\boldsymbol{x}$, and their corresponding class labels y. The final output is a strong classifier which takes the form

$$F_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{j=1}^{J} w_j h_j(\boldsymbol{x}). \tag{3}$$

Here $h_j(\cdot)$ is a base learner. From Theorem 2.1, we know that a matrix $\mathbf{X} \in \Gamma_1$ can be decomposed as

$$\mathbf{X} = \sum_{j=1}^{J} w_j \mathbf{Z}_j, \mathbf{Z}_j \in \Gamma_2. \tag{4}$$

By observing the similarity between Equations (3) and (4), we may view $\mathbf{Z}_j$ as a weak classifier and the matrix $\mathbf{X}$ as the strong classifier that we want to learn. This is exactly the problem that boosting methods have been designed to solve. This observation inspires us to solve a special type of semidefinite optimization problem using boosting techniques.

The sparse greedy approximation algorithm proposed by Zhang (2003) is an efficient method for solving a class of convex problems, and achieves fast convergence rates. It has also been shown that boosting algorithms can be interpreted within the general framework of (Zhang, 2003). The main idea of sequential greedy approximation, therefore, is as follows. Given an initialization $\boldsymbol{u}_0$, which is in a convex subset of a linear vector space, a matrix space or a functional space, the algorithm finds $\boldsymbol{u}_i$ and $\lambda \in (0,1)$ such that the objective function $F((1-\lambda)\boldsymbol{u}_{i-1} + \lambda \boldsymbol{u}_i)$ is minimized. Then the solution $\boldsymbol{u}_i$ is updated as $\boldsymbol{u}_i = (1-\lambda)\boldsymbol{u}_{i-1} + \lambda \boldsymbol{u}_i$ and the iteration goes on. Clearly, $\boldsymbol{u}_i$ must remain in the original space. As shown next, our first case, which learns a metric using the hinge loss, greatly resembles this idea.

2.4 Distance Metric Learning Using Proximity Comparison

The process of measuring distance using a Mahalanobis metric is equivalent to linearly transforming the data by a projection matrix $\mathbf{L} \in \mathbb{R}^{D \times d}$ (usually $D \geq d$) before calculating the standard Euclidean

Algorithm 1 The general framework of boosting.

Input: Training data.

1 Initialize a weight set $\boldsymbol{u}$ on the training examples;
2 **for** $j = 1, 2, \cdots,$ **do**
3 · Receive a weak hypothesis $h_j(\cdot)$;
4 · Calculate $w_j > 0$;
5 · Update $\boldsymbol{u}$.

Output: A convex combination of the weak hypotheses: $F_{\boldsymbol{w}}(\boldsymbol{x}) = \sum_{j=1}^{J} w_j h_j(\boldsymbol{x})$.

distance:

$$\mathbf{dist}_{ij}^2 = \|\mathbf{L}^\top \mathbf{a}_i - \mathbf{L}^\top \mathbf{a}_j\|_2^2 = (\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{L}\mathbf{L}^\top (\mathbf{a}_i - \mathbf{a}_j) = (\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{X}(\mathbf{a}_i - \mathbf{a}_j). \tag{5}$$

As described above, the problem of learning a Mahalanobis metric can be approached in terms of learning the matrix $\mathbf{L}$, or the p.s.d. matrix $\mathbf{X}$. If $\mathbf{X} = \mathbf{I}$, the Mahalanobis distance reduces to the Euclidean distance. If $\mathbf{X}$ is diagonal, the problem corresponds to learning a metric in which different features are given different weights, *a.k.a.*, feature weighting. Our approach is to learn a full p.s.d. matrix $\mathbf{X}$, however, using BOOSTMETRIC.

In the framework of large-margin learning, we want to maximize the distance between $\mathbf{dist}_{ij}$ and $\mathbf{dist}_{ik}$. That is, we wish to make $\mathbf{dist}_{ik}^2 - \mathbf{dist}_{ij}^2 = (\mathbf{a}_i - \mathbf{a}_k)^\top \mathbf{X}(\mathbf{a}_i - \mathbf{a}_k) - (\mathbf{a}_i - \mathbf{a}_j)^\top \mathbf{X}(\mathbf{a}_i - \mathbf{a}_j)$ as large as possible under some regularization. To simplify notation, we rewrite the distance between $\mathbf{dist}_{ij}^2$ and $\mathbf{dist}_{ik}^2$ as $\mathbf{dist}_{ik}^2 - \mathbf{dist}_{ij}^2 = \langle \mathbf{A}_r, \mathbf{X} \rangle$, where

$$\mathbf{A}_r = (\mathbf{a}_i - \mathbf{a}_k)(\mathbf{a}_i - \mathbf{a}_k)^\top - (\mathbf{a}_i - \mathbf{a}_j)(\mathbf{a}_i - \mathbf{a}_j)^\top, \tag{6}$$

for $r = 1, \cdots, |\mathcal{I}|$ and $|\mathcal{I}|$ is the size of the set of constraints $\mathcal{I}$ defined in Equation (2).

3. Algorithms

In this section, we define the optimization problems for metric learning. We mainly investigate the cases using the hinge loss, exponential loss and logistic loss functions. In order to derive an efficient optimization strategy, we look at their Lagrange dual problems and design boosting-like approaches for efficiency.

3.1 Learning with the Hinge Loss

Our goal is to derive a general algorithm for p.s.d. matrix learning with the hinge loss function. Assume that we want to find a p.s.d. matrix $\mathbf{X} \succcurlyeq 0$ such that a set of constraints

$$\langle \mathbf{A}_r, \mathbf{X} \rangle > 0, r = 1, 2, \cdots,$$

are satisfied as *well* as possible. Here $\mathbf{A}_r$ is as defined in (6). These constraints need not all be strictly satisfied and thus we define the margin $\rho_r = \langle \mathbf{A}_r, \mathbf{X} \rangle, \forall r$.

Putting it into the maximum margin learning framework, we want to minimize the following trace norm regularized objective function: $\sum_r F(\langle \mathbf{A}_r, \mathbf{X} \rangle) + v\,\mathbf{Tr}(\mathbf{X})$, with $F(\cdot)$ a convex loss function and v a regularization constant. Here we have used the trace norm regularization. Of course a

Frobenius norm regularization term can also be used here. Minimizing the Frobenius norm $||\mathbf{X}||_{\mathrm{F}}^2$, which is equivalent to minimize the ℓ_2 norm of the eigenvalues of $\mathbf{X}$, penalizes a solution that is far away from the identity matrix. With the hinge loss, we can write the optimization problem as:

$$\max_{\rho,\mathbf{X},\boldsymbol{\xi}} \rho - v\sum_{r=1}^{|\mathcal{I}|}\xi_r, \text{ s.t.: } \langle \mathbf{A}_r, \mathbf{X}\rangle \geq \rho - \xi_r, \forall r; \mathbf{X} \succcurlyeq 0, \mathbf{Tr}(\mathbf{X}) = 1; \boldsymbol{\xi} \geq \mathbf{0}. \tag{7}$$

Here $\mathbf{Tr}(\mathbf{X}) = 1$ removes the scale ambiguity because the distance inequalities are scale invariant.

We can decompose $\mathbf{X}$ into: $\mathbf{X} = \sum_{j=1}^{J} w_j \mathbf{Z}_j$, with $w_j > 0$, $\mathbf{Rank}(\mathbf{Z}_j) = 1$ and $\mathbf{Tr}(\mathbf{Z}_j) = 1$, $\forall j$. So we have

$$\langle \mathbf{A}_r, \mathbf{X}\rangle = \left\langle \mathbf{A}_r, \sum_{j=1}^{J} w_j \mathbf{Z}_j \right\rangle = \sum_{j=1}^{J} w_j \left\langle \mathbf{A}_r, \mathbf{Z}_j \right\rangle = \sum_{j=1}^{J} w_j \mathbf{H}_{rj} = \mathbf{H}_{r:}\boldsymbol{w}, \forall r. \tag{8}$$

Here $\mathbf{H}_{rj}$ is a shorthand for $\mathbf{H}_{rj} = \left\langle \mathbf{A}_r, \mathbf{Z}_j \right\rangle$. Clearly, $\mathbf{Tr}(\mathbf{X}) = \mathbf{1}^\top \boldsymbol{w}$. Using Theorem 2.1, we replace the p.s.d. conic constraint in the primal (7) with a linear convex combination of rank-one unitary matrices: $\mathbf{X} = \sum_j w_j \mathbf{Z}_j$, and $\mathbf{1}^\top \boldsymbol{w} = 1$. Substituting $\mathbf{X}$ in (7), we have

$$\max_{\rho,\boldsymbol{w},\boldsymbol{\xi}} \rho - v\sum_{r=1}^{|\mathcal{I}|}\xi_r, \text{ s.t.: } \mathbf{H}_{r:}\boldsymbol{w} \geq \rho - \xi_r, (r = 1, \ldots, |\mathcal{I}|); \boldsymbol{w} \geq \mathbf{0}, \mathbf{1}^\top \boldsymbol{w} = 1; \boldsymbol{\xi} \geq \mathbf{0}. \tag{9}$$

The Lagrange dual problem of the above linear programming problem (9) is easily derived:

$$\min_{\pi,\boldsymbol{u}} \pi \text{ s.t.: } \sum_{r=1}^{|\mathcal{I}|} u_r \mathbf{H}_{r:} \leq \pi \mathbf{1}^\top; \mathbf{1}^\top \boldsymbol{u} = 1, \mathbf{0} \leq \boldsymbol{u} \leq v\mathbf{1}. \tag{10}$$

We can then use column generation to solve the original problem iteratively by looking at both the primal and dual problems. See Shen et al. (2008) for the algorithmic details. In this work we are more interested in smooth loss functions such as the exponential loss and logistic loss, as presented in the sequel.

3.2 Learning with the Exponential Loss

By employing the exponential loss, we want to optimize

$$\begin{aligned} &\min_{\mathbf{X},\boldsymbol{\rho}} \log\left(\sum_{r=1}^{|\mathcal{I}|} \exp(-\rho_r)\right) + v\,\mathbf{Tr}(\mathbf{X}) \\ &\text{s.t.: } \rho_r = \langle \mathbf{A}_r, \mathbf{X}\rangle, r = 1, \cdots, |\mathcal{I}|, \ \mathbf{X} \succcurlyeq 0. \end{aligned} \tag{11}$$

Note that: 1) We are proposing a logarithmic version of the sum of exponential loss. This transform does not change the original optimization problem of sum of exponential loss because the logarithmic function is strictly monotonically increasing. 2) A regularization term $\mathbf{Tr}(\mathbf{X})$ has been applied. Without this regularization, one can always multiply $\mathbf{X}$ by an arbitrarily large scale factor in order to make the exponential loss approach zero in the case of all constraints being satisfied. This trace-norm regularization may also lead to low-rank solutions. 3) An auxiliary variable $\rho_r, r = 1, \ldots$ must be introduced for deriving a meaningful dual problem, as we show later.

We now derive the Lagrange dual of the problem that we are interested in. The original problem (11) now becomes

$$\begin{aligned} &\min_{\boldsymbol{\rho},\boldsymbol{w}} \log\left(\sum_{r=1}^{|\mathcal{I}|} \exp(-\rho_r)\right) + v\mathbf{1}^\top \boldsymbol{w} \\ &\text{s.t.: } \rho_r = \mathbf{H}_{r:}\boldsymbol{w}, r = 1, \cdots, |\mathcal{I}|; \boldsymbol{w} \geq \mathbf{0}. \end{aligned} \tag{12}$$

We have used the Equation (8). In order to derive its dual, we write its Lagrangian

$$L(\boldsymbol{w},\boldsymbol{\rho},\boldsymbol{u}) = \log\left(\sum_{r=1}^{|\mathcal{I}|}\exp(-\rho_r)\right) + \nu\mathbf{1}^\top\boldsymbol{w} + \sum_{r=1}^{|\mathcal{I}|}u_r(\rho_r - \mathbf{H}_{r:}\boldsymbol{w}) - \boldsymbol{p}^\top\boldsymbol{w}, \tag{13}$$

with $\boldsymbol{p} \geq 0$. The dual problem is obtained by finding the saddle point of L; *i.e.*, $\sup_{\boldsymbol{u}}\inf_{\boldsymbol{w},\boldsymbol{\rho}} L$.

$$\inf_{\boldsymbol{w},\boldsymbol{\rho}} L = \inf_{\boldsymbol{\rho}} \overbrace{\log\left(\sum_{r=1}^{|\mathcal{I}|}\exp(-\rho_r)\right) + \boldsymbol{u}^\top\boldsymbol{\rho}}^{L_1} + \inf_{\boldsymbol{w}} \overbrace{(\nu\mathbf{1}^\top - \sum_{r=1}^{|\mathcal{I}|}u_r\mathbf{H}_{r:} - \boldsymbol{p}^\top)\boldsymbol{w}}^{L_2} \tag{14}$$

$$= -\sum_{r=1}^{|\mathcal{I}|}u_r\log u_r. \tag{15}$$

The infimum of L_1 is found by setting its first derivative to zero and we have:

$$\inf_{\boldsymbol{\rho}} L_1 = \begin{cases} -\sum_r u_r\log u_r & \text{if } \boldsymbol{u} \geq \mathbf{0}, \mathbf{1}^\top\boldsymbol{u} = 1, \\ -\infty & \text{otherwise.} \end{cases} \tag{16}$$

The infimum is Shannon entropy. L_2 is linear in $\boldsymbol{w}$, hence it must be $\mathbf{0}$. It leads to

$$\sum_{r=1}^{|\mathcal{I}|}u_r\mathbf{H}_{r:} \leq \nu\mathbf{1}^\top. \tag{17}$$

The Lagrange dual problem of (12) is an entropy maximization problem, which writes

$$\max_{\boldsymbol{u}} -\sum_{r=1}^{|\mathcal{I}|}u_r\log u_r, \text{ s.t.: } \boldsymbol{u} \geq \mathbf{0}, \mathbf{1}^\top\boldsymbol{u} = 1, \text{and (17)}. \tag{18}$$

Weak and strong duality hold under mild conditions (Boyd and Vandenberghe, 2004). That means, one can usually solve one problem from the other. The KKT conditions link the optimal between these two problems. In our case, it is

$$u_r^\star = \frac{\exp(-\rho_r^\star)}{\sum_{k=1}^{|\mathcal{I}|}\exp(-\rho_k^\star)}, \forall r. \tag{19}$$

While it is possible to devise a totally-corrective column generation based optimization procedure for solving our problem as the case of LPBoost (Demiriz et al., 2002), we are more interested in considering *one-at-a-time* coordinate-wise descent algorithms, as the case of AdaBoost (Schapire, 1999). Let us start from some basic knowledge of column generation because our coordinate descent strategy is inspired by column generation.

If we know all the bases $\mathbf{Z}_j\,(j = 1\ldots J)$ and hence the entire matrix $\mathbf{H}$ is known. Then either the primal (12) or the dual (18) can be trivially solved (at least in theory) because both are convex optimization problems. We can solve them in polynomial time. Especially the primal problem is convex minimization with simple nonnegativeness constraints. Off-the-shelf software like LBFGS-B (Zhu et al., 1997) can be used for this purpose. Unfortunately, in practice, we do not access all the bases: the possibility of $\mathbf{Z}$ is infinite. In convex optimization, column generation is a technique that is designed for solving this difficulty.

Column generation was originally advocated for solving large scale linear programs (Lübbecke and Desrosiers, 2005). Column generation is based on the fact that for a linear program, the number of non-zero variables of the optimal solution is equal to the number of constraints. Therefore, although the number of possible variables may be large, we only need a small subset of these in

the optimal solution. For a general convex problem, we can use column generation to obtain an *approximate* solution. It works by only considering a small subset of the entire variable set. Once it is solved, we ask the question:"Are there any other variables that can be included to improve the solution?". So we must be able to solve the subproblem: given a set of dual values, one either identifies a variable that has a favorable reduced cost, or indicates that such a variable does not exist. Essentially, column generation finds the variables with negative reduced costs without explicitly enumerating all variables.

Instead of directly solving the primal problem (12), we find the most violated constraint in the dual (18) iteratively for the current solution and adds this constraint to the optimization problem. For this purpose, we need to solve

$$\hat{\mathbf{Z}} = \text{argmax}_{\mathbf{Z}} \left\{ \sum_{r=1}^{|\mathcal{I}|} u_r \langle \mathbf{A}_r, \mathbf{Z} \rangle, \text{ s.t.: } \mathbf{Z} \in \Psi_1 \right\}. \tag{20}$$

We discuss how to efficiently solve (20) later. Now we move on to derive a coordinate descent optimization procedure.

3.3 Coordinate Descent Optimization

We show how an AdaBoost-like optimization procedure can be derived.

3.3.1 OPTIMIZING FOR w_j

Since we are interested in the *one-at-a-time* coordinate-wise optimization, we keep $w_1, w_2, \ldots, w_{j-1}$ fixed when solving for w_j. The cost function of the primal problem is (in the following derivation, we drop those terms irrelevant to the variable w_j)

$$C_p(w_j) = \log\left[\sum_{r=1}^{|\mathcal{I}|} \exp(-\rho_r^{j-1}) \cdot \exp(-\mathbf{H}_{rj} w_j)\right] + v w_j.$$

Clearly, C_p is convex in w_j and hence there is only one minimum that is also globally optimal. The first derivative of C_p w.r.t. w_j vanishes at optimality, which results in

$$\sum_{r=1}^{|\mathcal{I}|} (\mathbf{H}_{rj} - v) u_r^{j-1} \exp(-w_j \mathbf{H}_{rj}) = 0. \tag{21}$$

If $\mathbf{H}_{rj}$ is discrete, such as $\{+1, -1\}$ in standard AdaBoost, we can obtain a closed-form solution similar to AdaBoost. Unfortunately in our case, $\mathbf{H}_{rj}$ can be any real value. We instead use bisection to search for the optimal w_j. The bisection method is one of the root-finding algorithms. It repeatedly divides an interval in half and then selects the subinterval in which a root exists. Bisection is a simple and robust, although it is not the fastest algorithm for root-finding. Algorithm 2 gives the bisection procedure. We have utilized the fact that the l.h.s. of (21) must be positive at w_l. Otherwise no solution can be found. When $w_j = 0$, clearly the l.h.s. of (21) is positive.

3.3.2 UPDATING $\boldsymbol{u}$

The rule for updating $\boldsymbol{u}$ can be easily obtained from (19). At iteration j, we have

$$u_r^j \propto \exp(-\rho_r^j) \propto u_r^{j-1} \exp(-\mathbf{H}_{rj} w_j), \text{ and } \sum_{r=1}^{|\mathcal{I}|} u_r^j = 1,$$

Algorithm 2 Bisection search for w_j.

Input: An interval $[w_l, w_u]$ known to contain the optimal value of w_j and convergence tolerance $\varepsilon > 0$.

```
repeat
    · w_j = 0.5(w_l + w_u);
    · if l.h.s. of (21) > 0 then
        w_l = w_j;
    else
        w_u = w_j.
until w_u − w_l < ε ;
```

Output: w_j.

derived from (19). So once w_j is calculated, we can update $\boldsymbol{u}$ as

$$u_r^j = \frac{u_r^{j-1}\exp(-\mathbf{H}_{rj}w_j)}{z}, r = 1,\ldots,|\mathcal{I}|, \tag{22}$$

where z is a normalization factor so that $\sum_{r=1}^{|\mathcal{I}|} u_r^j = 1$. This is exactly the same as AdaBoost.

3.4 The Base Learning Algorithm

In this section, we show that the optimization problem (20) can be exactly and efficiently solved using eigenvalue-decomposition (EVD).

From $\mathbf{Z} \succcurlyeq 0$ and $\mathbf{Rank}(\mathbf{Z}) = 1$, we know that $\mathbf{Z}$ has the format: $\mathbf{Z} = \boldsymbol{v}\boldsymbol{v}^\top$, $\boldsymbol{v} \in \mathbb{R}^D$; and $\mathbf{Tr}(\mathbf{Z}) = 1$ means $\|\boldsymbol{v}\|_2 = 1$. We have

$$\left\langle \sum_{r=1}^{|\mathcal{I}|} u_r \mathbf{A}_r, \mathbf{Z} \right\rangle = \boldsymbol{v}\left(\sum_{r=1}^{|\mathcal{I}|} u_r \mathbf{A}_r\right)\boldsymbol{v}^\top.$$

By denoting

$$\hat{\mathbf{A}} = \sum_{r=1}^{|\mathcal{I}|} u_r \mathbf{A}_r, \tag{23}$$

the base learning optimization equals:

$$\max_{\boldsymbol{v}} \boldsymbol{v}^\top \hat{\mathbf{A}} \boldsymbol{v}, \text{ s.t.: } \|\boldsymbol{v}\|_2 = 1. \tag{24}$$

It is clear that the largest eigenvalue of $\hat{\mathbf{A}}$, $\lambda_{\max}(\hat{\mathbf{A}})$, and its corresponding eigenvector $\boldsymbol{v}_1$ gives the solution to the above problem. Note that $\hat{\mathbf{A}}$ is symmetric.

$\lambda_{\max}(\hat{\mathbf{A}})$ is also used as one of the stopping criteria of the algorithm. Form the condition (17), $\lambda_{\max}(\hat{\mathbf{A}}) < v$ means that we are not able to find a new base matrix $\hat{\mathbf{Z}}$ that violates (17)—the algorithm converges.

Eigenvalue decompositions is one of the main computational costs in our algorithm. There are approximate eigenvalue solvers, which guarantee that for a symmetric matrix $\mathbf{U}$ and any $\varepsilon > 0$, a vector $\boldsymbol{v}$ is found such that $\boldsymbol{v}^\top \mathbf{U} \boldsymbol{v} \geq \lambda_{\max} - \varepsilon$. To approximately find the largest eigenvalue and eigenvector can be very efficient using Lanczos or power method. We can use the MATLAB function eigs to calculate the largest eigenvector, which calls mex files of ARPACK. ARPACK is a collection of Fortran subroutines designed to solve large scale eigenvalue problems. When the

Algorithm 3 Positive semidefinite matrix learning with stage-wise boosting.

Input:

- Training set triplets $(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \in \mathcal{I}$; Compute $\mathbf{A}_r, r = 1, 2, \cdots$, using (6).
- J: maximum number of iterations;
- (optional) regularization parameter v; We may simply set v to a very small value, *e.g.*, 10^{-7}.

Initialize: $u_r^0 = \frac{1}{|\mathcal{I}|}, r = 1 \cdots |\mathcal{I}|$;
for $j = 1, 2, \cdots, J$ **do**
  · Find a new base $\mathbf{Z}_j$ by finding the largest eigenvalue ($\lambda_{\max}(\hat{\mathbf{A}})$) and its eigenvector of $\hat{\mathbf{A}}$ in (23);
  · **if** $\lambda_{\max}(\hat{\mathbf{A}}) < v$ **then**
    break (converged);
  · Compute w_j using Algorithm 2;
  · Update $\boldsymbol{u}$ to obtain $u_r^j, r = 1, \cdots |\mathcal{I}|$ using (22);

Output: The final p.s.d. matrix $\mathbf{X} \in \mathbb{R}^{D \times D}$, $\mathbf{X} = \sum_{j=1}^{J} w_j \mathbf{Z}_j$.

input matrix is symmetric, this software uses a variant of the Lanczos process called the implicitly restarted Lanczos method.

Another way to reduce the time for computing the leading eigenvector is to compute an approximate EVD by a fast Monte Carlo algorithm such as the linear time SVD algorithm developed in (Drineas et al., 2004).

We summarize our main algorithmic results in Algorithm 3.

3.5 Learning with the Logistic Loss

We have considered the exponential loss in the last content. The proposed framework is so general that it can also accommodate other convex loss functions. Here we consider the logistic loss, which penalizes mis-classifications with more moderate penalties than the exponential loss. It is believed on noisy data, the logistic loss may achieve better classification performance.

With the same settings as in the case of the exponential loss, we can write our optimization problem as

$$\begin{aligned} &\min_{\boldsymbol{\rho}, \boldsymbol{w}} \textstyle\sum_{r=1}^{|\mathcal{I}|} \mathrm{logit}(\rho_r) + v\mathbf{1}^\top \boldsymbol{w} \\ &\text{s.t.:} \rho_r = \mathbf{H}_{r:}\boldsymbol{w}, r = 1, \cdots, |\mathcal{I}|, \boldsymbol{w} \geq 0. \end{aligned} \tag{25}$$

Here $\mathrm{logit}(\cdot)$ is the logistic loss defined as $\mathrm{logit}(z) = \log(1 + \exp(-z))$. Similarly, we derive its Lagrange dual as

$$\begin{aligned} &\min_{\boldsymbol{u}} \textstyle\sum_{r=1}^{|\mathcal{I}|} \mathrm{logit}^*(-u_r) \\ &\text{s.t.:} \textstyle\sum_{r=1}^{|\mathcal{I}|} u_r \mathbf{H}_{r:} \leq v\mathbf{1}^\top, \end{aligned} \tag{26}$$

where $\text{logit}^*(\cdot)$ is the Fenchel conjugate function of $\text{logit}(\cdot)$, defined as

$$\text{logit}^*(-u) = u\log(u) + (1-u)\log(1-u), \tag{27}$$

when $0 \leq u \leq 1$, and ∞ otherwise. So the Fenchel conjugate of $\text{logit}(\cdot)$ is the binary entropy function. We have reversed the sign of $\boldsymbol{u}$ when deriving the dual.

Again, according to the KKT conditions, we have

$$u_r^\star = \frac{\exp(-\rho_r^\star)}{1+\exp(-\rho_r^\star)}, \ \forall r, \tag{28}$$

at optimality. From (28) we can also see that u must be in $(0,1)$.

Similarly, we want to optimize the primal cost function in a coordinate descent way. First, let us find the relationship between u_r^j and u_r^{j-1}. Here j is the iteration index. From (28), it is trivial to obtain

$$u_r^j = \frac{1}{(1/u_r^{j-1}-1)\exp(\mathbf{H}_{rj}w_j)+1}, \ \forall r. \tag{29}$$

The optimization of w_j can be solved by looking for the root of

$$\textstyle\sum_{r=1}^{|\mathcal{I}|}\mathbf{H}_{rj}u_r^j - v = 0, \tag{30}$$

where u_r^j is a function of w_j as defined in (29).

Therefore, in the case of the logistic loss, to find w_j, we modify the bisection search of Algorithm 2:

- Line 3: **if** l.h.s. of (30) > 0 **then** ...

and Line 7 of Algorithm 3:

- Line 7: Update $\boldsymbol{u}$ using (29).

3.6 Totally Corrective Optimization

In this section, we derive a totally-corrective version of BOOSTMETRIC, similar to the case of TotalBoost (Warmuth et al., 2006; Shen and Li, 2010) for classification, in the sense that the coefficients of all weak learners are updated at each iteration.

Unlike the stage-wise optimization, here we do not need to keep previous weights of weak learners $w_1, w_2, \ldots, w_{j-1}$. Instead, the weights of all the selected weak learners $w_1, w_2, \ldots, w_j$ are updated at each iteration j. As discussed, our learning procedure is able to employ various loss functions such as the hinge loss, exponential loss or logistic loss. To devise a totally-corrective optimization procedure for solving our problem efficiently, we need to ensure the object function to be differentiable with respect to the variables $w_1, w_2, \ldots, w_j$. Here, we use the exponential loss function and the logistic loss function. It is possible to use sub-gradient descent methods when a non-smooth loss function like the hinge loss is used.

It is clear that solving for $\boldsymbol{w}$ is a typical convex optimization problem since it has a differentiable and convex function (12) when the exponential loss is used, or (25) when the logistic loss is used. Hence it can be solved using off-the-shelf gradient-descent solvers like L-BFGS-B (Zhu et al., 1997).

Algorithm 4 Positive semidefinite matrix learning with totally corrective boosting.

Input:

- Training set triplets $(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \in \mathcal{I}$; Compute $\mathbf{A}_r, r = 1, 2, \cdots$, using (6).
- J: maximum number of iterations;
- Regularization parameter v.

Initialize: $u_r^0 = \frac{1}{|\mathcal{I}|}, r = 1 \cdots |\mathcal{I}|$;
for $j = 1, 2, \cdots, J$ **do**
  · Find a new base $\mathbf{Z}_j$ by finding the largest eigenvalue ($\lambda_{\max}(\hat{\mathbf{A}})$) and its eigenvector of $\hat{\mathbf{A}}$ in (23);
  · **if** $\lambda_{\max}(\hat{\mathbf{A}}) < v$ **then**
    break (converged);
  · Optimize for $w_1, w_2, \cdots, w_j$ by solving the primal problem (12) when the exponential loss is used or (25) when the logistic loss is used;
  · Update $\boldsymbol{u}$ to obtain $u_r^j, r = 1, \cdots |\mathcal{I}|$ using (19) (exponential loss) or (28) (logistic loss);

Output: The final p.s.d. matrix $\mathbf{X} \in \mathbb{R}^{D \times D}$, $\mathbf{X} = \sum_{j=1}^{J} w_j \mathbf{Z}_j$.

Since all the weights $w_1, w_2, \ldots, w_j$ are updated, u_r^j on $r = 1 \ldots |\mathcal{I}|$ need not to be updated but re-calculated at each iteration j. To calculate u_r^j, we use (19) (exponential loss) or (28) (logistic loss) instead of (22) or (29) respectively. Totally-corrective BOOSTMETRIC methods are very simple to implement. Algorithm 4 gives the summary of this algorithm. Next, we show the convergence property of Algorithm 4. Formally, we want to show the following theorem.

Theorem 3.1. *Algorithm 4 makes progress at each iteration. In other words, the objective value is decreased at each iteration. Therefore, in the limit, Algorithm 4 solves the optimization problem* (12) *(or* (25)*) globally to a desired accuracy.*

Proof. Let us consider the exponential loss case of problem (12). The proof follows the same discussion for the logistic loss, or any other smooth convex loss function. Assume that the current solution is a finite subset of base learners (rank-one trace-one matrices) and their corresponding linear coefficients $\boldsymbol{w}$. If we add a base matrix $\hat{\mathbf{Z}}$ that is not in the current subset, and the corresponding $\hat{w} = 0$, then the objective value and the solution must remain unchanged. We can conclude that the current learned base learners and $\boldsymbol{w}$ are the optimal solution already.

Consider the case that this optimality condition is violated. We need to show that we can find a base learner $\hat{\mathbf{Z}}$, which is not in the current set of all the selected base learners, such that $\hat{w} > 0$ holds. Now assume that $\hat{\mathbf{Z}}$ is the base learner found by solving (24), and the convergence condition $\lambda_{\max}(\hat{\mathbf{A}}) \leq v$ is not satisfied. So, we have $\lambda_{\max}(\hat{\mathbf{A}}) = \left\langle \sum_{r=1}^{|\mathcal{I}|} u_r \mathbf{A}_r, \hat{\mathbf{Z}} \right\rangle > v$.

If, after this weak learner $\hat{\mathbf{Z}}$ is added into the primal problem, the primal solution remains unchanged, i.e., the corresponding $\hat{w} = 0$, then from the optimality condition that L_2 in (14) must be zero, we know that $\hat{p} = v - \left\langle \sum_{r=1}^{|\mathcal{I}|} u_r \mathbf{A}_r, \hat{\mathbf{Z}} \right\rangle < 0$. This contradicts the fact the Lagrange multiplier $\hat{p} \geq 0$.

We can conclude that after the base learner $\hat{\mathbf{Z}}$ is added into the primal problem, its corresponding $\hat{w}$ must admit a positive value. It means that one more free variable is added into the problem and re-solving the primal problem would reduce the objective value. Hence a strict decrease in the objective is guaranteed. So Algorithm 4 makes progress at each iteration.

Furthermore, as the optimization problems involved are all convex, there are no local optimal solutions. Therefore Algorithm 4 is guaranteed to converge to the global solution.

Note that the above proof establishes the convergence of Algorithm 4 but it remains unclear about the convergence rate. ■

3.7 Multi-pass BOOSTMETRIC

In this section, we show that BOOSTMETRIC can use multi-pass learning to enhance the performance.

Our BOOSTMETRIC uses training set triplets $(\mathbf{a}_i, \mathbf{a}_j, \mathbf{a}_k) \in \mathcal{I}$ as input for training. The Mahalanobis distance metric $\mathbf{X}$ can be viewed as a linear transformation in the Euclidean space by projecting the data using matrix $\mathbf{L}$ ($\mathbf{X} = \mathbf{L}\mathbf{L}^\top$). That is, nearest neighbors of samples using Mahalanobis distance metric $\mathbf{X}$ are the same as nearest neighbors using Euclidean distance in the transformed space. BOOSTMETRIC assumes that the triplets of input training set approximately represent the actual nearest neighbors of samples in the transformed space defined by the Mahalanobis metric. However, even though the triplets of BOOSTMETRIC consist of nearest neighbors of the original training samples, generated triplets are not exactly the same as the actual nearest neighbors of training samples in the transformed space by $\mathbf{L}$.

We can refine the results of BOOSTMETRIC iteratively, as in the multiple-pass LMNN (Weinberger and Saul, 2009): BOOSTMETRIC can estimate the triplets in the transformed space under a multiple-pass procedure as close to actual triplets as possible. The rule for multi-pass BOOSTMETRIC is simple. At each pass p ($p = 1, 2, \cdots$), we decompose the learned Mahalanobis distance metric $\mathbf{X}_{p-1}$ of previous pass into transformation matrix $\mathbf{L}_p$. The initial matrix $\mathbf{L}_1$ is an identity matrix. Then we generate the training set triplets from the set of points $\{\mathbf{L}^\top\mathbf{a}_1, \ldots, \mathbf{L}^\top\mathbf{a}_m\}$ where $\mathbf{L} = \mathbf{L}_1 \cdot \mathbf{L}_2 \cdots \mathbf{L}_p$. The final Mahalanobis distance metric $\mathbf{X}$ becomes $\mathbf{L}\mathbf{L}^\top$ in Multi-pass BOOSTMETRIC.

4. Experiments

In this section, we present experiments on data visualization, classification and image retrieval tasks.

4.1 An Illustrative Example

We demonstrate a data visualization problem on an artificial toy dataset (concentric circles) in Fig. 1. The dataset has four classes. The first two dimensions follow concentric circles while the left eight dimensions are all random Gaussian noise. In this experiment, 9000 triplets are generated for training. When the scale of the noise is large, PCA fails find the first two informative dimensions. LDA fails too because clearly each class does not follow a Gaussian distraction and their centers overlap at the same point. The proposed BOOSTMETRIC algorithm find the informative features. The eigenvalues of $\mathbf{X}$ learned by BOOSTMETRIC are $\{0.542, 0.414, 0.007, 0, \cdots, 0\}$, which indicates that BOOSTMETRIC successfully reveals the data's underlying 2D structure. We have used the exponential loss in this experiment.

Table 1: Comparison of test classification error rates (%) of a 3-nearest neighbor classifier on benchmark datasets. Results of NCA are not available either because the algorithm does not converge or due to the out-of-memory problem. BoostMetric-E indicates BOOSTMETRIC with the exponential loss and BoostMetric-L is BOOSTMETRIC with the logistic loss; both use stage-wise optimization. "MP" means Multiple-Pass BOOSTMETRIC and "TC" is BOOSTMETRIC with totally corrective optimization. We report computational time as well.

		MNIST	USPS	Letters	yFaces	bal	wine	iris
# of samples		70,000	11,000	20,000	2,414	625	178	150
# of triplets		450,000	69,300	94,500	15,210	3,942	1,125	945
dimension		784	256	16	1,024	4	13	4
dimension after PCA		164	60		300			
# of samples for training		50,000	7,700	10,500	1,690	438	125	105
# cross validation samples		10,000	1,650	4,500	362	94	27	23
# test samples		10,000	1,650	5,000	362	93	26	22
# of classes		10	10	26	38	3	3	3
# of runs		1	10	1	10	10	10	10
Error Rates	Euclidean	3.19	4.78 (0.40)	5.42	28.07 (2.07)	18.60 (3.96)	28.08 (7.49)	3.64 (4.18)
	PCA	3.10	3.49 (0.62)		28.65 (2.18)			
	LDA	8.76	6.96 (0.68)	4.44	**5.08 (1.15)**	12.58 (2.38)	0.77 (1.62)	3.18 (3.07)
	RCA	7.85	5.35 (0.52)	4.64	7.65 (1.08)	17.42 (3.58)	**0.38 (1.22)**	3.18 (3.07)
	NCA					18.28 (3.58)	28.08 (7.49)	3.18 (3.74)
	LMNN	2.30	3.49 (0.62)	3.82	14.75 (12.11)	12.04 (5.59)	3.46 (3.82)	3.64 (2.87)
	ITML	2.80	3.85 (1.13)	7.20	19.39 (2.11)	10.11 (4.06)	28.46 (8.35)	3.64 (3.59)
	BoostMetric-E	2.65	2.53 (0.47)	3.06	6.91 (1.90)	10.11 (3.45)	3.08 (3.53)	3.18 (3.74)
	BoostMetric-E, MP	2.62	2.24 (0.40)	2.80	6.77 (1.77)	10.22 (4.43)	1.92 (2.03)	3.18 (4.31)
	BoostMetric-E, TC	2.20	2.25 (0.51)	2.82	7.13 (1.40)	10.22 (2.39)	4.23 (3.82)	3.18 (3.07)
	BoostMetric-E, MP, TC	2.34	2.23 (0.34)	3.74	7.29 (1.58)	10.32 (3.09)	2.69 (3.17)	3.18 (4.31)
	BoostMetric-L	2.66	2.38 (0.31)	2.80	6.93 (1.59)	9.89 (3.12)	3.08 (3.03)	3.18 (3.74)
	BoostMetric-L, MP	2.72	2.22 (0.31)	2.70	6.66 (1.35)	10.22 (4.25)	1.15 (1.86)	3.18 (4.31)
	BoostMetric-L, TC	**2.10**	**2.13 (0.41)**	2.48	7.71 (1.68)	9.57 (3.18)	3.85 (4.05)	3.64 (2.87)
	BoostMetric-L, MP, TC	2.11	2.10 (0.42)	**2.36**	7.15 (1.32)	**8.49 (3.71)**	3.08 (3.03)	**2.73 (2.35)**
Comp. Time	LMNN	10.98h	20s	1249s	896s	5s	2s	2s
	ITML	0.41h	72s	55s	5970s	8s	4s	4s
	BoostMetric-E	2.83h	144s	3s	628s	less than 1s	2s	less than 1s
	BoostMetric-L	0.89h	65s	34s	256s	less than 1s	2s	less than 1s

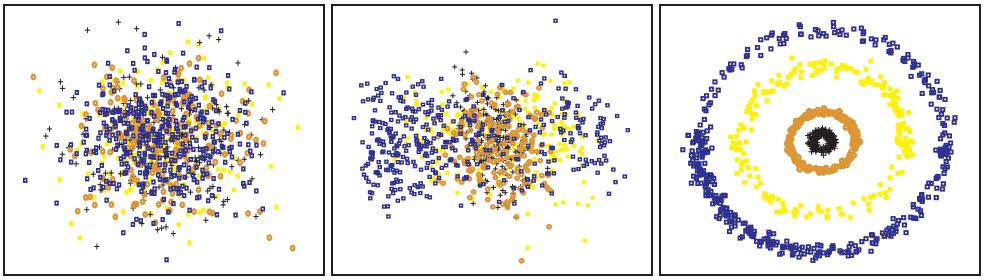

Figure 1: The data are projected into 2D with PCA (left), LDA (middle) and BOOSTMETRIC (right). Both PCA and LDA fail to recover the data structure. The local structure of the data is preserved after projection by BOOSTMETRIC.

4.2 Classification on Benchmark Datasets

We evaluate BOOSTMETRIC on 7 datasets of different sizes. Some of the datasets have very high dimensional inputs. We use PCA to decrease the dimensionality before training on these datasets (MNIST, USPS and yFaces). PCA pre-processing helps to eliminate noises and speed up computation. Table 1 summarizes the datasets in detail. We have used USPS and MNIST handwritten digits, Yale face recognition datasets, and a few UCI machine learning datasets[2] .

Experimental results are obtained by averaging over 10 runs (except for large datasets MNIST and Letter). We randomly split the datasets for each run. We have used the same mechanism to generate training triplets as described in (Weinberger et al., 2005). Briefly, for each training point $\mathbf{a}_i$, k nearest neighbors that have same labels as y_i (targets), as well as k nearest neighbors that have different labels from y_i (imposers) are found. We then construct triplets from $\mathbf{a}_i$ and its corresponding targets and imposers. For all the datasets, we have set $k = 3$ (3-nearest-neighbor). We have compared our method against a few methods: RCA (Bar-Hillel et al., 2005), NCA (Goldberger et al., 2004), ITML (Davis et al., 2007) and LMNN (Weinberger et al., 2005). Also in Table 1, "Euclidean" is the baseline algorithm that uses the standard Euclidean distance. The codes for these compared algorithms are downloaded from the corresponding author's website. Experiment setting for LMNN follows (Weinberger et al., 2005). The slack variable parameter for ITML is tuned using cross validation over the values $0.01, 0.1, 1, 10$ as in (Davis et al., 2007). For BOOSTMETRIC, we have set $v = 10^{-7}$, the maximum number of iterations $J = 500$.

BOOSTMETRIC has different variants which use 1) the exponential loss (BOOSTMETRIC-E), 2) the logistic loss (BOOSTMETRIC-L), 3) multiple pass evaluation (MP) for updating triplets with the exponential and logistic loss, and 4) two optimization strategies, namely, stage-wise optimization and totally corrective optimization. The experiments are conducted by using Matlab and a C-mex implementation of the L-BFGS-B algorithm.

As reported in Table 1, we can conclude: 1) BOOSTMETRIC consistently improves the accuracy of kNN classification using Euclidean distance on most datasets. So learning a Mahalanobis metric based upon the large margin concept indeed leads to improvements in kNN classification. 2) BOOSTMETRIC outperforms other state-of-the-art algorithms in most cases (on 5 out of 7 datasets). LMNN is the second best algorithm on these 7 data sets statistically. LMNN's results are consistent

2. http://archive.ics.uci.edu/ml/

v	10^{-8}	10^{-7}	10^{-6}	10^{-5}	10^{-4}
Bal	8.98 (2.59)	8.88 (2.52)	8.88 (2.52)	8.88 (2.52)	8.93 (2.52)
B-Cancer	2.11 (0.69)	2.11 (0.69)	2.11 (0.69)	2.11 (0.69)	2.11 (0.69)
Diabetes	26.0 (1.33)	26.0 (1.33)	26.0 (1.33)	26.0 (1.34)	26.0 (1.46)

Table 2: Test error (%) of a 3-nearest neighbor classifier with different values of the parameter v. Each experiment is run 10 times. We report the mean and variance. As expected, as long as v is sufficiently small, in a wide range it almost does not affect the final classification performance.

with those given in (Weinberger et al., 2005). ITML is faster than BOOSTMETRIC on most large datasets such as MNIST. However it has higher error rates than BOOSTMETRIC in our experiment. 3) NCA can only be run on a few small data sets. In general NCA does not perform well. Initialization is important for NCA because NCA's objective function is highly non-convex and can only find a local optimum.

In this experiment, LMNN solves for the global optimum (learning $\mathbf{X}$) except for the Wine dataset. When the LMNN solver solves for $\mathbf{X}$ on the Wine dataset, the error rate is large ($20.77\% \pm 14.18\%$). So instead we have solved for the projection matrix $\mathbf{L}$ on Wine. Also note that the number of training data on Iris, Wine and Bal in (Weinberger et al., 2005) are different from our experiment. We have used these datasets from UCI. For the experiment on MNIST, if we deskew the handwritten digits data first as in (Weinberger and Saul, 2009), the final accuracy can be slightly improved. Here we have not deskewed the data.

4.2.1 INFLUENCE OF v

Previously, we claim that the stage-wise version of BOOSTMETRIC is parameter-free like AdaBoost. However, we do have a parameter v. Actually, AdaBoost simply set $v = 0$. The coordinate-wise gradient descent optimization strategy of AdaBoost leads to an ℓ_1-norm regularized maximum margin classifier (Rosset et al., 2004). It is shown that AdaBoost minimizes its loss criterion with an ℓ_1 constraint on the coefficient vector. Given the similarity of the optimization of BOOSTMETRIC with AdaBoost, we conjecture that BOOSTMETRIC has the same property. Here we empirically prove that *as long as v is sufficiently small, the final performance is not affected by the value of v*. We have set v from 10^{-8} to 10^{-4} and run BOOSTMETRIC on 3 UCI datasets. Table 2 reports the final 3NN classification error with different v. The results are nearly identical.

For the totally corrective version of BOOSTMETRIC, similar results are observed. Actually for LMNN, it was also reported that the regularization parameter does not have a significant impact on the final results in a wide range (Weinberger and Saul, 2009).

4.2.2 COMPUTATIONAL TIME

As we discussed, one major issue in learning a Mahalanobis distance is heavy computational cost because of the semidefiniteness constraint.

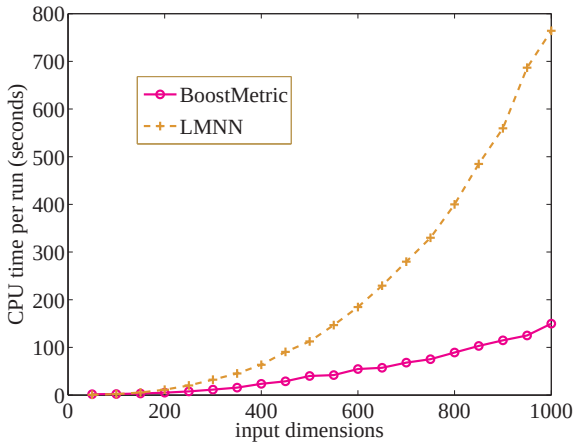

Figure 2: Computation time of the proposed BOOSTMETRIC (stage-wise, exponential loss) and the LMNN method versus the input data's dimensions on an artificial dataset. BOOSTMETRIC is faster than LMNN with large input dimensions because at each iteration BOOSTMETRIC only needs to calculate the largest eigenvector and LMNN needs a full eigen-decomposition.

We have shown the running time of the proposed algorithm in Table 1 for the classification tasks[3]. Our algorithm is generally fast. Our algorithm involves matrix operations and an EVD for finding its largest eigenvalue and its corresponding eigenvector. The time complexity of this EVD is $O(D^2)$ with D the input dimensions. We compare our algorithm's running time with LMNN in Fig. 2 on the artificial dataset (concentric circles). Our algorithm is stage-wise BOOSTMETRIC with the exponential loss. We vary the input dimensions from 50 to 1000 and keep the number of triplets fixed to 250. LMNN does not use standard interior-point SDP solvers, which do not scale well. Instead LMNN heuristically combines sub-gradient descent in both the matrices $\mathbf{L}$ and $\mathbf{X}$. At each iteration, $\mathbf{X}$ is projected back onto the p.s.d. cone using EVD. So a full EVD with time complexity $O(D^3)$ is needed. Note that LMNN is much faster than SDP solvers like CSDP (Borchers, 1999). As seen from Fig. 2, when the input dimensions are low, BOOSTMETRIC is comparable to LMNN. As expected, when the input dimensions become large, BOOSTMETRIC is significantly faster than LMNN. Note that our implementation is in Matlab. Improvements are expected if implemented in C/C++.

4.3 Visual Object Categorization

In the following experiments, unless otherwise specified, BOOSTMETRIC means the stage-wise BOOSTMETRIC with the exponential loss.

The proposed BOOSTMETRIC and the LMNN are further compared on visual object categorization tasks. The first experiment uses four classes of the Caltech-101 object recognition database (Fei-Fei et al., 2006), including Motorbikes (798 images), Airplanes (800), Faces (435),

3. We have run all the experiments on a desktop with an Intel Core^{TM}2 Duo CPU, 4G RAM and Matlab 7.7 (64-bit version).

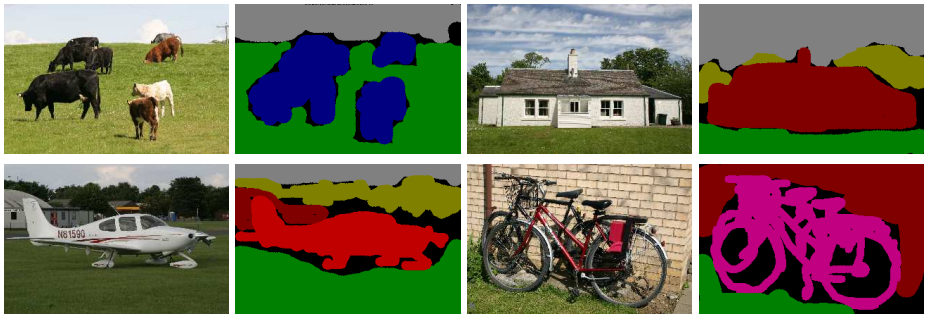

Figure 3: Examples of the images in the MSRC data set and the pre-segmented regions labeled using different colors.

and Background-Google (520). The task is to label each image according to the presence of a particular object. This experiment involves both object categorization (Motorbikes versus Airplanes) and object retrieval (Faces versus Background-Google) problems. In the second experiment, we compare the two methods on the MSRC data set including 240 images[4]. The objects in the images can be categorized into nine classes, including *building, grass, tree, cow, sky, airplane, face, car and bicycle*. Different from the first experiment, each image in this database often contains multiple objects. The regions corresponding to each object have been manually pre-segmented, and the task is to label each region according to the presence of a particular object. Some examples are shown in Fig. 3.

4.3.1 EXPERIMENT ON THE CALTECH-101 DATASET

For each image of the four classes, a number of interest regions are identified by the Harris-affine detector (Mikolajczyk and Schmid, 2004) and each region is characterized by the SIFT descriptor (Lowe, 2004). The total number of interest regions extracted from the four classes are about $134,000$, $84,000$, $57,000$, and $293,000$, respectively. To accumulate statistics, the images of two involved object classes are randomly split as 10 pairs of training/test subsets. Restricted to the images in a training subset (those in a test subset are only used for test), their local descriptors are clustered to form visual words by using k-means clustering. Each image is then represented by a histogram containing the number of occurrences of each visual word.

Motorbikes versus Airplanes This experiment discriminates the images of a motorbike from those of an airplane. In each of the 10 pairs of training/test subsets, there are 959 training images and 639 test images. Two visual codebooks of size 100 and 200 are used, respectively. With the resulting histograms, the proposed BOOSTMETRIC and the LMNN are learned on a training subset and evaluated on the corresponding test subset. Their averaged classification error rates are compared in Fig. 4 (left). For both visual codebooks, the proposed BOOSTMETRIC achieves lower error rates than the LMNN and the Euclidean distance, demonstrating its superior performance. We also apply a linear SVM classifier with its regularization parameter carefully tuned by 5-fold cross-validation. Its error rates are $3.87\% \pm 0.69\%$ and $3.00\% \pm 0.72\%$ on the two visual codebooks, respectively. In contrast, a 3NN with BOOSTMETRIC has error rates $3.63\% \pm 0.68\%$ and $2.96\% \pm$

4. See `http://research.microsoft.com/en-us/projects/objectclassrecognition/`.

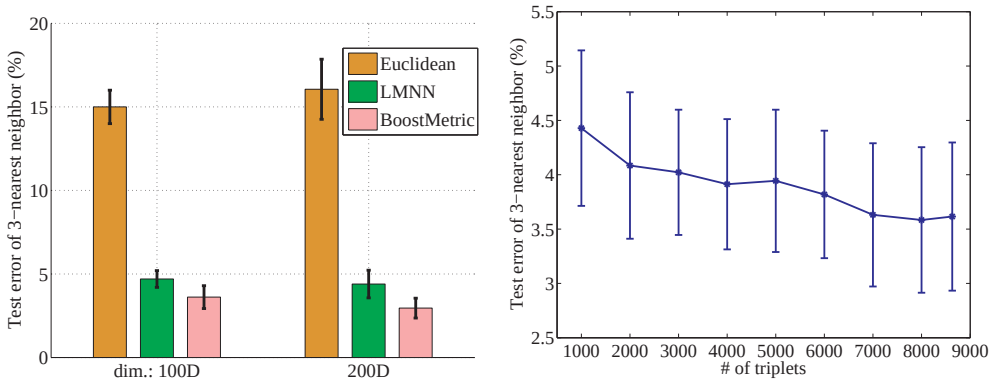

Figure 4: Test error (3-nearest neighbor) of BOOSTMETRIC on the Motorbikes versus Airplanes datasets. The second plot shows the test error against the number of training triplets with a 100-word codebook.

0.59%. Hence, the performance of the proposed BOOSTMETRIC is comparable to the state-of-the-art SVM classifier. Also, Fig. 4 (right) plots the test error of the BOOSTMETRIC against the number of triplets for training. The general trend is that more triplets lead to smaller errors.

Faces versus Background-Google This experiment uses the two object classes as a retrieval problem. The target of retrieval is face images. The images in the class of Background-Google are randomly collected from the Internet and they represent the non-target class. BOOSTMETRIC is first learned from a training subset and retrieval is conducted on the corresponding test subset. In each of the 10 training/test subsets, there are 573 training images and 382 test images. Again, two visual codebooks of size 100 and 200 are used. Each face image in a test subset is used as a query, and its distances from other test images are calculated by the proposed BoostMetric, LMNN and the Euclidean distance, respectively. For each metric, the *Precision* of the retrieved top 5, 10, 15 and 20 images are computed. The *Precision* values from each query are averaged on this test subset and then averaged over the 10 test subsets. The retrieval precision of these metrics is shown in Fig. 5 (with a codebook size 100). As we can see that the BOOSTMETRIC consistently attains the highest values on both visual codebooks, which again verifies its advantages over LMNN and Euclidean distance. With a codebook size 200, very similar results are obtained.

4.3.2 EXPERIMENT ON THE MSRC DATASET

The 240 images of the MSRC database are randomly halved into 10 groups of training and test sets. Given a set of training images, the task is to predict the class label for each of the pre-segmented regions in a test image. We follow the work in (Winn et al., 2005) to extract features and conduct experiments. Specifically, each image is converted from the RGB color space to the CIE Lab color space. First, three Gaussian low-pass filters are applied to the L, a, and b channels, respectively. The standard deviation σ of the filters are set to 1, 2, and 4, respectively, and the filter size is defined as 4σ. This step produces 9 filter responses for each pixel in an image. Second, three Laplacian of Gaussian (LoG) filters are applied to the L channel only, with $\sigma = 1, 2, 4, 8$ and the filter size of 4σ. This step gives rise to 4 filter responses for each pixel. Lastly, the first derivatives of the Gaussian filter with $\sigma = 2, 4$ are computed from the L channel along the row and column directions,

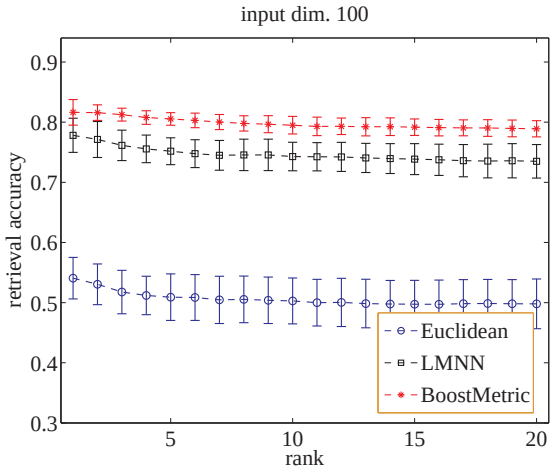

Figure 5: Retrieval accuracy of distance metric learning algorithms on the Faces versus Background-Google dataset. Error bars show the standard deviation.

respectively. This results in 4 more filter responses. After applying this set of filter banks, each pixel is represented by a 17-dimensional feature vectors. All the feature vectors from a training set are clustered using the k-means clustering with a Mahalanobis distance[5]. By setting k to 2000, a visual codebook of 2000 visual words is obtained. We implement the word-merging approach in (Winn et al., 2005) and obtain a compact and discriminative codebook of 300 visual words. Each pre-segmented object region is then represented as a 300-dimensional histogram.

The proposed BOOSTMETRIC is compared with the LMNN algorithm as follows. With 10 nearest neighbors information, about $20,000$ triplets are constructed and used to train the BOOSTMETRIC. To ensure convergence, the maximum number of iterations is set as 5000 in the optimization of training BOOSTMETRIC. The training of LMNN follows the default setting. kNN classifiers with the two learned Mahalanobis distances and the Euclidean distance are applied to each training and test group to categorize an object region. The categorization error rate on each test group is summarized in Table 3. As expected, both learned Mahalanobis distances achieve superior categorization performance to the Euclidean distance. Moreover, the proposed BOOSTMETRIC achieves better performance than the LMNN, as indicated by its lower average categorization error rate and the smaller standard deviation. Also, the kNN classifier using the proposed BOOSTMETRIC achieves comparable or even higher categorization performance than those reported in (Winn et al., 2005). Besides the categorization performance, we compare the computational efficiency of the BOOSTMETRIC and the LMNN in learning a Mahalanobis distance. The computational time result is based on the Matlab codes for both methods. In this experiment, the average time cost by the BOOSTMETRIC for learning the Mahalanobis distance is 3.98 hours, whereas the LMNN takes about 8.06 hours to complete this process. Hence, the proposed BOOSTMETRIC has a shorter training process than the LMNN method. This again demonstrates the computational advantage of the BOOSTMETRIC over the LMNN method.

5. Note that this Mahalanobis distance is different from the one that we are going to learn with the BOOSTMETRIC.

group index	Euclidean	LMNN	BOOSTMETRIC
1	9.19	6.71	4.59
2	5.78	3.97	3.25
3	6.69	2.97	2.60
4	5.54	3.69	4.43
5	6.52	5.80	4.35
6	7.30	4.01	3.28
7	7.75	2.21	2.58
8	7.20	4.17	4.55
9	6.13	3.07	4.21
10	8.42	5.13	5.86
average:	7.05	4.17	**3.97**
standard devision:	1.16	1.37	**1.03**

Table 3: Comparison of the categorization performance.

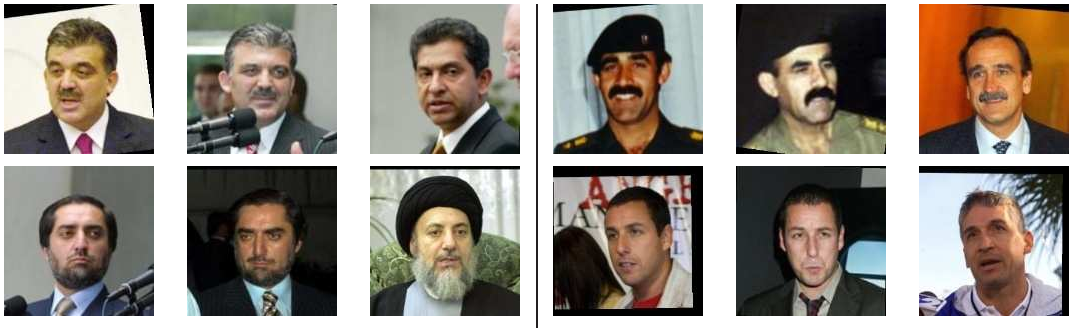

Figure 6: Four generated triplets based on the pairwise information provided by the LFW data set. For the three images in each triplet, the first two belong to the same individual and the third one is a different individual.

4.4 Unconstrained Face Recognition

We use the "labeled faces in the wild" (LFW) dataset (Huang et al., 2007) for face recognition in this experiment.

This is a data set of unconstrained face images, which has a large range of variations seen in real world, including $13,233$ images of $5,749$ people collected from news articles on Internet. The face recognition task here is *pair matching*—given two face images, to determine if these two images are of the same individual. So we classify unseen pairs to determine whether each image in the pair indicates the same individual or not, by applying MkNN of (Guillaumin et al., 2009) instead of kNN.

Features of face images are extracted by computing 3-scale, 128-dimensional SIFT descriptors (Lowe, 2004), which center on 9 points of facial features extracted by a facial feature descriptor, same as described in (Guillaumin et al., 2009). PCA is then performed on the SIFT vectors to reduce the dimension to between 100 and 400.

Simple recognition systems with a single descriptor Table 4 shows our BOOSTMETRIC's performance by varying PCA dimensionality and the number of triplets. Increasing the number of

number of triplets	100D	200D	300D	400D
3,000	80.91 (1.76)	82.39 (1.73)	83.40 (1.46)	83.64 (1.66)
6,000	81.13 (1.76)	82.59 (1.84)	83.58 (1.25)	83.70 (1.73)
9,000	81.01 (1.69)	82.63 (1.68)	83.65 (1.70)	83.72 (1.47)
12,000	81.06 (1.63)	83.00 (1.38)	83.60 (1.89)	83.57 (1.47)
15,000	81.10 (1.71)	82.78 (1.83)	83.69 (1.62)	83.80 (1.85)
18,000	81.37 (2.15)	83.19 (1.76)	83.60 (1.66)	83.81 (1.55)

Table 4: Comparison of the face recognition accuracy (%) of our proposed BOOSTMETRIC on the LFW dataset by varying the PCA dimensionality and the number of triplets for each fold.

training triplets gives slight improvement of recognition accuracy. The dimension after PCA has more impact on the final accuracy for this task.

In Fig. 7, we have drawn ROC curves of other algorithms for face recognition. To obtain our ROC curve, M*k*NN has moved the threshold value across the distributions of match and mismatch similarity scores. Fig. 7 (a) shows methods that use a single descriptor and a single classifier only. As can be seen, our system using BOOSTMETRIC outperforms all the others in the literature with a very small computational cost.

Complex recognition systems with one or more descriptors Fig. 7 (b) plots the performance of more complicated recognition systems that use hybrid descriptors or combination of classifiers. See Table 5 for details. We can see that the performance of our BOOSTMETRIC is close to the state-of-the-art.

In particular, BOOSTMETRIC outperforms the method of (Guillaumin et al., 2009), which has a similar pipeline but uses LMNN for learning a metric. This comparison also confirms the importance of learning an appropriate metric for vision problems.

5. Conclusion

We have presented a new algorithm, BOOSTMETRIC, to learn a positive semidefinite metric using boosting techniques. We have generalized AdaBoost in the sense that the weak learner of BOOSTMETRIC is a matrix, rather than a classifier. Our algorithm is simple and efficient. Experiments show its better performance over a few state-of-the-art existing metric learning methods. We are currently combining the idea of on-line learning into BOOSTMETRIC to make it handle even larger data sets.

We also want to learn a metric using BOOSTMETRIC in the semi-supervised, and multi-task learning setting. It has been shown in (Weinberger and Saul, 2009) that the classification performance can be improved by learning multiple local metrics. We will extend BOOSTMETRIC to learn multiple metrics. Finally, we will explore to generalize BOOSTMETRIC for solving more general semidefinite matrix learning problems in machine learning.

References

A. Bar-Hillel, T. Hertz, N. Shental, and D. Weinshall. Learning a Mahalanobis metric from equivalence constraints. *J. Machine Learning Research*, 6:937–965, 2005.

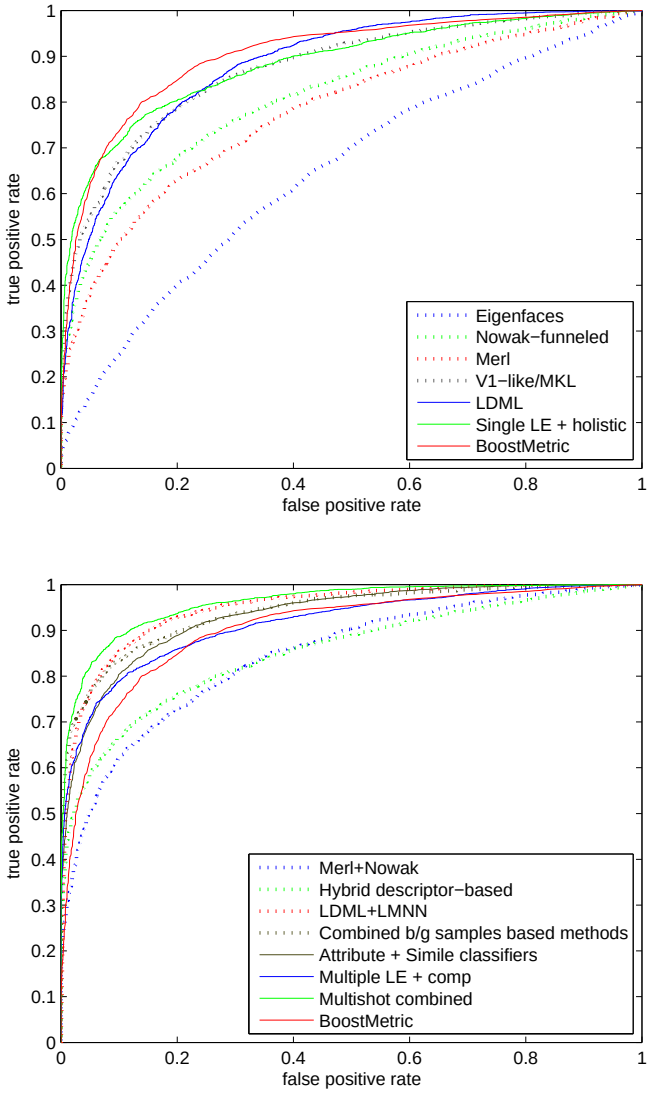

Figure 7: (top) ROC Curves that use a single descriptor and a single classifier, (bottom) ROC curves that use hybrid descriptors are plotted. Our BOOSTMETRIC with a single classifier is also plotted. Each point on the curves is the average over the 10 folds of rates for a fixed threshold.

O. Boiman, E. Shechtman, and M. Irani. In defense of nearest-neighbor based image classification. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2008.

B. Borchers. CSDP, a C library for semidefinite programming. *Optimization Methods and Software*, 11(1):613–623, 1999.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

	single descriptor + single classifier	multiple descriptors/classifiers
(Turk and Pentland, 1991)	60.02 (0.79) 'Eigenfaces'	-
(Nowak and Jurie, 2007)	73.93 (0.49) 'Nowak-funneled'	-
(Huang et al., 2008)	70.52 (0.60) 'Merl'	76.18 (0.58) 'Merl+Nowak'
(Wolf et al., 2008)	-	78.47 (0.51) 'Hybrid descriptor-based'
(Wolf et al., 2009)	72.02 -	86.83 (0.34) 'Combined b/g samples based'
(Pinto et al., 2009)	79.35 (0.55) 'V1-like/MKL'	-
(Taigman et al., 2009)	83.20 (0.77) -	**89.50 (0.40)** 'Multishot combined'
(Kumar et al., 2009)	-	85.29 (1.23) 'attribute + simile classifiers'
(Cao et al., 2010)	81.22 (0.53) 'single LE + holistic'	84.45 (0.46) 'multiple LE + comp'
(Guillaumin et al., 2009)	83.2 (0.4) 'LDML'	87.5 (0.4) 'LMNN + LDML'
BOOSTMETRIC	**83.81 (1.55)** 'BOOSTMETRIC' on SIFT	-

Table 5: Test accuracy in percentage (mean and standard deviation) on the LFW dataset. ROC curve labels in Fig. 7 are described here with details.

Z. Cao, Q. Yin, X. Tang, and J. Sun. Face recognition with learning-based descriptor. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2010.

G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operation Research*, 8 (1):101–111, 1960.

J. V. Davis, B. Kulis, P. Jain, S. Sra, and I. S. Dhillon. Information-theoretic metric learning. In *Int'l Conf. Machine Learning*, pages 209–216, Corvalis, Oregon, 2007. ACM Press.

A. Demiriz, K.P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1-3):225–254, 2002.

P. Drineas, R. Kannan, and M. Mahoney. Fast Monte Carlo algorithms for matrices II: Computing a compressed approximate matrix decomposition. *SIAM J. Computing*, 36:2006, 2004.

L. Fei-Fei, R. Fergus, and P. Perona. One-shot learning of object categories. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 28(4):594–611, April 2006.

P. A. Fillmore and J. P. Williams. Some convexity theorems for matrices. *Glasgow Math. Journal*, 12:110–117, 1971.

A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *Proc. IEEE Int'l Conf. Computer Vision*, 2007.

A. Globerson and S. Roweis. Metric learning by collapsing classes. In *Proc. Advances in Neural Information Processing Systems*, 2005.

J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood component analysis. In *Proc. Advances in Neural Information Processing Systems*. MIT Press, 2004.

M. Guillaumin, J. Verbeek, and C. Schmid. Is that you? metric learning approaches for face identification. In *Proc. IEEE Int'l Conf. Computer Vision*, 2009.

T. Hastie and R. Tibshirani. Discriminant adaptive nearest neighbor classification. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 18(6):607–616, 1996.

X. He, S. Yan, Y. Hu, P. Niyogi, and H.-J. Zhang. Face recognition using Laplacianfaces. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 27(3):328–340, 2005.

G. B. Huang, M. Ramesh, T. Berg, and E. Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst, October 2007.

G. B. Huang, M. J. Jones, and E. Learned-Miller. LFW results using a combined nowak plus merl recognizer. In *Faces in Real-Life Images Workshop in Euro. Conf. Computer Vision*, 2008.

B. Jian and B. C. Vemuri. Metric learning using Iwasawa decomposition. In *Proc. IEEE Int'l Conf. Computer Vision*, Rio de Janeiro, Brazil, 2007. IEEE.

M. Krein and D. Milman. On extreme points of regular convex sets. *Studia Mathematica*, 9:133–138, 1940.

N. Kumar, A. C. Berg, P. N. Belhumeur, and S. K. Nayar. Attribute and simile classifiers for face verification. In *Proc. IEEE Int'l Conf. Computer Vision*, 2009.

D. G. Lowe. Distinctive image features from scale-invariant keypoints. *Int'l J. Computer Vision*, 60 (2):91–110, 2004.

M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operation Research*, 53 (6):1007–1023, 2005.

K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *Int'l J. Computer Vision*, 60(1):63–86, 2004.

E. Nowak and F. Jurie. Learning visual similarity measures for comparing never seen objects. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2007.

M. L. Overton and R. S. Womersley. On the sum of the largest eigenvalues of a symmetric matrix. *SIAM J. Matrix Analysis and Application*, 13(1):41–45, 1992.

N. Pinto, J. J. DiCarlo, and D. D. Cox. How far can you get with a modern face recognition test set using only simple features? In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 2009.

R. Rosales and G. Fung. Learning sparse metrics via linear programming. In *Proc. ACM SIGKDD Int'l Conf. Knowledge discovery and Data Mining*, pages 367–373. ACM, 2006.

S. Rosset, J. Zhu, and T. Hastie. Boosting as a regularized path to a maximum margin classifier. *J. Machine Learning Research*, 5:941–973, 2004.

R. E. Schapire. Theoretical views of boosting and applications. In *Proc. Int'l Conf. Algorithmic Learning Theory*, pages 13–25, London, UK, 1999. Springer-Verlag.

M. Schultz and T. Joachims. Learning a distance metric from relative comparisons. In *Proc. Advances in Neural Information Processing Systems*. MIT Press, 2003.

C. Shen and H. Li. On the dual formulation of boosting algorithms. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32(12):2216–2231, 2010.

C. Shen, A. Welsh, and L. Wang. PSDBoost: Matrix-generation linear programming for positive semidefinite matrices learning. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Proc. Advances in Neural Information Processing Systems*, pages 1473–1480, Vancouver, B.C., Canada, December 2008. MIT Press.

C. Shen, J. Kim, L. Wang, and A. van den Hengel. Positive semidefinite metric learning with boosting. In Y. Bengio, D. Schuurmans, J. Lafferty, C. Williams, and A. Culotta, editors, *Proc. Advances in Neural Information Processing Systems*, pages 1651–1659, Vancouver, B.C., Canada, December 2009. MIT Press.

Y. Taigman, L. Wolf, and T. Hassner. Multiple one-shots for utilizing class label information. In *Proc. British Machine Vision Conf.*, 2009.

K. Tsuda, G. Rätsch, and M. K. Warmuth. Matrix exponentiated gradient updates for on-line learning and Bregman projection. *J. Machine Learning Research*, 6:995–1018, December 2005.

M. A. Turk and A. P. Pentland. Face recognition using eigenfaces. In *Proc. IEEE Int'l Conf. Computer Vision and Pattern Recognition*, 1991.

H. Wang, H. Huang, and C. Ding. Discriminant Laplacian embedding. In *Proc. AAAI Conf. Artificial Intelligence*, pages 618–623, 2010a.

S. Wang and R. Jin. An information geometry approach for distance metric learning. In *Proc. Int'l Conf. Artificial Intelligence and Statistics*, pages 591–598, 2009.

Z. Wang, Y. Hu, and L.-T. Chia. Image-to-class distance metric learning for image classification. In *Proc. Euro. Conf. Computer Vision*, volume Lecture Notes in Computer Science 6311/2010, pages 706–719, 2010b.

M. K. Warmuth, J. Liao, and G. Rätsch. Totally corrective boosting algorithms that maximize the margin. In *Int'l Conf. Machine Learning*, pages 1001–1008, Pittsburgh, Pennsylvania, 2006.

K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *Int'l J. Computer Vision*, 70(1):77–90, 2006.

K. Q. Weinberger and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. *J. Machine Learning Research*, 10:207–244, 2009.

K. Q. Weinberger, J. Blitzer, and L. K. Saul. Distance metric learning for large margin nearest neighbor classification. In *Proc. Advances in Neural Information Processing Systems*, pages 1473–1480, 2005.

J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proc. IEEE Int'l Conf. Computer Vision*, pages 1800–1807, 2005.

L. Wolf, T. Hassner, and Y. Taigman. Descriptor based methods in the wild. In *Faces in Real-Life Images Workshop in Euro. Conf. Computer Vision*, 2008.

L. Wolf, T. Hassner, and Y. Taigman. Similarity scores based on background samples. In *Proc. Asian Conf. Computer Vision*, 2009.

E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Proc. Advances in Neural Information Processing Systems*. MIT Press, 2002.

J. Yu, J. Amores, N. Sebe, P. Radeva, and Q. Tian. Distance learning for similarity estimation. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 30(3):451–462, 2008.

T. Zhang. Sequential greedy approximation for certain convex optimization problems. *IEEE Trans. Information Theory*, 49(3):682–691, 2003.

C. Zhu, R. H. Byrd, and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Trans. Mathematical Software*, 23(4):550–560, 1997.